



PATENT

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

Title: "A Client-Server Computing System Capable of Validating Cached Data
Based on Data Transformation"

Applicants: Edlund et al.

Attorney Docket No.: ARC920030019US1

Serial No.: 10/789,743

Examiner: Binh Van Ho

Filed: February 27, 2004

Art Unit: 2163

5

Mail Stop: Board of Patent Appeals and Interferences
Commissioner for Patents
P.O.Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

This appeal brief is submitted under 35 U.S.C. §134. This appeal is further to Appellants'

10 Notice of Appeal filed July 12, 2007.

Table of Contents

<u>Section</u>	<u>Title</u>	<u>Page</u>
(1)	Real Party in Interest	2
(2)	Related Appeals and Interferences	2
(3)	Status of Claims	2
(4)	Status of Amendments	2
(5)	Summary of Claimed Subject Matter	2
(6)	Grounds of Rejection to be Reviewed on Appeal	6
(7)	Argument	6
App. A	Claims Appendix	23
App. B	Evidence Appendix	29
App. C	Related Proceedings Appendix	30

(1) Real Party in Interest

The real party in interest is International Business Machines Corporation.

(2) Related Appeals and Interferences

No other appeals or interferences exist that relate to the present application or appeal.

(3) Status of Claims

Claims 1-27 are pending and remain in the application. By the Final Office Action dated March 21, 2007, the Examiner has rejected claims 1-27 under 35 U.S.C. § 103(a) as being unpatentable over Lipkin (U.S. 2005/0154699) (hereinafter "Lipkin") in view of Challenger (U.S. Patent No. 6,026,413) (hereinafter "Challenger"). All pending claims and all of the rejections are hereby appealed. A copy of the appealed claims is enclosed herewith as Appendix A.

(4) Status of Amendments

No amendments are outstanding.

(5) Summary of Claimed Subject Matter

Independent Claim 1

Independent claim 1 relates to a client-server system capable of validating cached eXtensible Markup Language (XML) data, where the system includes (1) a data store for storing XML data, (2) a server for retrieving and updating XML data in the data store to service client requests, (3) a transformation engine for transforming XML data into a format suitable for a client application based on a set of transformation rules, (4) a cache for temporarily storing transformed XML data as data objects for later reuse, (5) a cache monitor for ensuring that cached objects are validated when changes to XML data in the data store are detected by the server, and (6) an object dependency mapper for automatically and continuously determining dependencies between XML data in the data store and sets of

transformation rules. (Please see Application as Filed, page 4, lines 22-31 and page 8, line 11 to page 9, line 33.)

Independent Claim 16

Independent claim 10 relates to, in a client-sever computing system having a cache
5 and storing eXtensible Markup Language (XML) data as data objects, a method for
determining invalid cached objects, where the method includes (1) transforming XML data
into a format suitable for a client application based on a set of transformation rules, (2)
determining dependencies between cached objects and XML data related to the cached
objects, (3) monitoring updates to the related XML data, and (4) determining the cached
10 objects that are affected by changes to the related XML data based on the dependencies.
(Please see Application as Filed, page 4, lines 22-31, page 8, line 11 to page 9, line 33, and
page 12, lines 21-32.)

Independent Claim 22

Independent claim 5 relates to, in a client-sever computing system having a cache and
15 storing eXtensible Markup Language (XML) data as data objects, a computer program
product for determining invalid cached objects, where the computer program product includes
(1) a computer readable medium, (2) means, provided on the computer readable medium, for
transforming XML data into a format suitable for a client application based on a set of
transformation rules, (3) means, provided on the computer readable medium, for determining
20 dependencies between cached objects and XML data related to the cached objects, (4) means,
provided on the computer readable medium, for monitoring updates to the related XML data,
and (5) means, provided on the computer readable medium, for determining the cached
objects that are affected by changes to the related XML data based on the dependencies.
(Please see Application as Filed, page 4, lines 22-31, page 7, lines 13-18, page 8, line 11 to
25 page 9, line 33, and page 12, lines 21-32.)

The means plus function “means, provided on the computer readable medium, for
transforming XML data into a format suitable for a client application based on a set of
transformation rules” of claim 22 corresponds to the following structure, material, or acts
described in the specification as corresponding to the claimed function:

“A transformation engine 35 transforms data in the data store 32 into a suitable format to be presented to a client 30, such as the hypertext markup language (HTML). The transformation is performed based on a set of transformation rules 36. Each transformation rule refers to a subset of data in the data store 32. A group of the transformation rules in combination with static presentation information is referred to as a style sheet. For example, a style sheet can be used to generate an HTML page containing both dynamic bank account information from a data store and static presentation information using HTML tags. The transformation engine 35 thus receives a style sheet and data from the data store 32 as input, and outputs a data object that has been transformed using the style sheet. In the preferred embodiment of the invention, the transformation rules are implemented using the eXtensible Stylesheet language (XSL) and include multiple XPath expressions. The XPath expressions refer to portions of the XML data in the data store 32 are referenced by these XPath expressions. The XPath expressions are used to determine how a particular style sheet depends on the underlying XML data, as described below in reference to the object dependency mapper.” (Please see Application as Filed, page 8, lines 32 to page 9, line 18.)

The means plus function “means, provided on the computer readable medium, for determining dependencies between cached objects and XML data related to the cached objects” of claim 22 corresponds to the following structure, material, or acts described in the specification as corresponding to the claimed function:

(a) “A cache monitor 34 communicates with the cache 33 to ensure that the cached data objects are still valid in view of recent updates to the underlying data in the data store 32. The cache monitor 34 receives XML data update requests from a server application running on the server 31 in the form of XML fragments. An XML fragment consists of an XPath expression that identifies the data in the data store 32 to be updated, inserted or deleted.

Further details on the preferred embodiment of the cache monitor 34 are described later in the specification.” (Please see Application as Filed, page 8, lines 25-31.);

(b) “At step 61, the system determines the dependencies between the data objects currently in the cache memory and their underlying data in the data store by accessing the table of dependencies.” (Please see Application as Filed, page 12, lines 27-29.); and

(c) “At step 71, the dependencies between the data objects in the cache and their corresponding data in the data store are automatically identified and maintained in a dependency table by the transformation rule alert service.” (Please see Application as Filed, page 13, lines 9-12.)

5 The means plus function “means, provided on the computer readable medium, for monitoring updates to the related XML data” of claim 22 corresponds to the following structure, material, or acts described in the specification as corresponding to the claimed function:

10 (a) “A cache monitor 34 communicates with the cache 33 to ensure that the cached data objects are still valid in view of recent updates to the underlying data in the data store 32. The cache monitor 34 receives XML data update requests from a server application running on the server 31 in the form of XML fragments. An XML fragment consists of an XPath expression that identifies the data in the data store 32 to be updated, inserted or deleted. Further details on the preferred embodiment of the
15 cache monitor 34 are described later in the specification.” (Please see Application as Filed, page 8, lines 25-31.);

(b) “At step 62, the system monitors updates to the underlying data that might affect the cached objects. ” (Please see Application as Filed, page 12, lines 29-30.); and

20 (c) “At step 72, the invention identifies the tree nodes that are affected by data updates in the data store. This is done by examining each data update to identify the relevant XPath expressions and the nodes referenced by these XPath expressions.”. (Please see Application as Filed, page 13, lines 12-15.)

25 The means plus function “means, provided on the computer readable medium, for determining the cached objects that are affected by changes to the related XML data based on the dependencies” of claim 22 corresponds to the following structure, material, or acts described in the specification as corresponding to the claimed function:

(a) “The system includes the additional components object manager 38 and transformation rule alert service 39 for improved performance. The object manager 38 generates new data objects for the cache 33 and refreshes the data objects currently in the

cache 33 in response to requests for data from the server 31. The object manager 38 constructs the new data objects when needed, for example, as a result of a data transformation. In addition, whenever the server 31 issues a data refresh request, the object manager 38 immediately rebuilds the relevant cached data objects. Optionally, the object manager 38 periodically examines the cached data objects and refreshes those objects that have expired, i.e., those that have been indicated as being invalid by the cache monitor 34. To decide how frequently a data object should be refreshed, the object manager 38 can either use the profile information associated with each type of cached data objects or use statistics gathered by a server application. The frequently used data objects in the cache 33 would be refreshed more often by the object manager 38 while infrequently accessed objects are automatically removed from the cache 33 by the object manager 38.” (Please see Application as Filed, page 11, lines 13-28.);

(b) “At step 63, the data objects in the cache that are affected by the data updates are determined based on the dependencies.” (Please see Application as Filed, page 12, lines 30-32.); and

(c) “At step 75, the cached data objects that have been transformed by these style sheets are indicated as being invalid by the object manager due to updates in the data store.”. (Please see Application as Filed, page 13, lines 18-20.)

(6) Grounds of Rejection to be Reviewed on Appeal

The issue for review is whether claims 1-27 are unpatentable over Lipkin in view of Challenger under 35 U.S.C. § 103(a).

(7) Argument

A. Introduction

The issue for review is whether claims 1-27 are unpatentable over Lipkin in view of Challenger under 35 U.S.C. § 103(a).

B. Whether claims 1-27 unpatentable over Lipkin in view of Challenger under 35 U.S.C. § 103(a)

Applicants respectfully traverse the obviousness rejection of claims 1-27 over Lipkin in view of Challenger under 35 U.S.C. § 103(a), and submit that claims 1-27 are not obvious over Lipkin in view of Challenger under 35 U.S.C. § 103(a), and are patentable thereover. In support of this position, Applicants submit the following argument.

5 **1. Legal Standards for Obviousness**

The following legal authorities set the general standards in support of Applicant's position of non obviousness, with emphasis added for added clarity:

- 10 • MPEP §2143.03, "All Claim Limitations Must Be Taught or Suggested: To establish prima facie obviousness of a claimed invention, **all the claim limitations must be taught or suggested by the prior art**. In re Royka, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). **"All words in a claim must be considered** in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim
15 depending therefrom is nonobvious. In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)."
- 20 • MPEP §2143.01, "The Prior Art Must Suggest The Desirability Of The Claimed Invention: There are three possible sources for a motivation to combine references: the nature of the problem to be solved, the teachings of the prior art, and the knowledge of persons of ordinary skill in the art." In re Rouffet, 149 F.3d 1350, 1357, 47 USPQ2d 1453, 1457-58 (Fed. Cir. 1998) (**The combination of the references taught every element of the claimed invention, however without a motivation to combine, a rejection based on a prima facie case of obvious was held improper.**). The level of skill in the art
25 cannot be relied upon to provide the suggestion to combine references. Al-Site Corp. v. VSI Int'l Inc., 174 F.3d 1308, 50 USPQ2d 1161 (Fed. Cir. 1999).
- 30 • "**Obviousness cannot be established** by combining the teachings of the prior art to produce the claimed invention, **absent some teaching or suggestion** supporting the combination." In re Fine, 837 F.2d at 1075, 5 USPQ2d at 1598 (citing ACS Hosp. Sys. v. Montefiore Hosp., 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984)). **What a reference teaches** and whether it teaches toward or **away from the claimed invention**
35 are questions of fact. See Raytheon Co. v. Roper Corp., 724 F.2d 951, 960-61, 220 USPQ 592, 599-600 (Fed. Cir. 1983), cert. denied, 469 U.S. 835, 83 L. Ed. 2d 69, 105 S. Ct. 127 (1984)."
- 40 • "When a rejection depends on a combination of prior art references, there must be **some teaching, suggestion, or motivation** to combine the references. See In re Geiger, 815 F.2d 686, 688, 2 USPQ2d 1276, 1278 (Fed. Cir. 1987). **Obviousness can only be established by combining or modifying** the teachings of the prior art to produce the

claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See MPEP 2143.01; In re Kotzab, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000); In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); and In re Jones, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

- “With respect to core factual findings in a determination of patentability, however, the Board cannot simply reach conclusions based on its own understanding or experience -- or on its assessment of what would be basic knowledge or common sense. Rather, the Board must point to some concrete evidence in the record in support of these findings.” See In re Zurko, 258 F.3d 1379 (Fed. Cir. 2001).
- “We have noted that evidence of a suggestion, teaching, or motivation to combine may flow from the prior art references themselves, the knowledge of one of ordinary skill in the art, or, in some cases, from the nature of the problem to be solved, see Pro-Mold & Tool Co. v. Great Lakes Plastics, Inc., 75 F.3d 1568, 1573, 37 USPQ2d 1626, 1630 (Fed. Cir. 1996), Para-Ordinance Mfg. v. SGS Imports Intern., Inc., 73 F.3d 1085, 1088, 37 USPQ2d 1237, 1240 (Fed. Cir. 1995), although “the suggestion more often comes from the teachings of the pertinent references,” Rouffet, 149 F.3d at 1355, 47 USPQ2d at 1456. The range of sources available, however, does not diminish the requirement for actual evidence. That is, the showing must be clear and particular. See, e.g., C.R. Bard, 157 F.3d at 1352, 48 USPQ2d at 1232. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not “evidence.” E.g., McElmurry v. Arkansas Power & Light Co., 995 F.2d 1576, 1578, 27 USPQ2d 1129, 1131 (Fed. Cir. 1993) (“Mere denials and conclusory statements, however, are not sufficient to establish a genuine issue of material fact.”); In re Sichert, 566 F.2d 1154, 1164, 196 USPQ 209, 217 (CCPA 1977).” See In re Dembiczak, 175 F. 3d 994 (Fed. Cir. 1999).
- “To prevent the use of hindsight based on the invention to defeat patentability of the invention, this court requires the examiner to show a motivation to combine the references that create the case of obviousness. In other words, the examiner must show reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed.” See In re Rouffet, 149, F.3d 1350 (Fed. Cir. 1998).
- The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. In re Mills, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Although a prior art device “may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so.” 916 F.2d at 682, 16 USPQ2d at 1432.). See also In re Fritch, 972 F.2d 1260, 23 USPQ2d 1780

(Fed. Cir. 1992) (flexible landscape edging device which is conformable to a ground surface of varying slope not suggested by combination of prior art references).

- If the **proposed modification would render the prior art invention being modified unsatisfactory** for its intended purpose, then there is no suggestion or motivation to make the proposed modification. In re Gordon, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).

2. Application of the Obviousness Standard to the Present Invention

By the Office Action dated March 21, 2007, the Examiner has rejected claims 1-27 under 35 U.S.C. § 103(a) as being unpatentable over Lipkin in view of Challenger. In order to form a proper obviousness rejection of a claim under 35 U.S.C. § 103(a), a collection of references together must teach or suggest each element of the claim, including the relationships between the elements. If any element is not fully taught by the combined references, the rejection cannot be sustained.

Evaluating Lipkin in view of Challenger in this light, it is appropriate to examine the portions of Lipkin in view of Challenger that the Examiner has pointed to as teaching the claimed elements of the rejected claims.

Claims 1-15

The Examiner asserted that, for claim 1,

Lipkin discloses substantially all of the elements, a client-server system capable of validating cached eXtensible Markup Language (XML) data comprising a data store for storing XML data; a server for retrieving and updating XML data in the data store to service client requests; a transformation engine for transforming XML data into a format suitable for a client application based on a set of transformation rules (Paragraph [1190])

(See Office Action, page 2.) The Examiner then admitted that, for claim 1,

Lipkin discloses substantially all of the elements . . . except
a cache for temporarily storing transformed XML data as data
objects for later reuse;
5 a cache monitor for ensuring that cached objects are validated
when changes to XML data in the data store are detected by the
server; and
an object dependency mapper for automatically and
continuously determining dependencies between XML data in
10 the data store and sets of transformation rules.

(See Office Action, page 2.) The Examiner then asserted that, for claim 1,

Challenger teaches in figures 1, 4-16, 18-21, 23-28, and 30-42,
15 a cache for temporarily store, and
monitor the change to data,
an object dependency mapper for automatically and
continuously determining dependencies between data in store
and set of transformation rules (Abstract; col. 4, lines 6-10;
20 col. 10, lines 14-24; col. 29, lines 32 +).

(See Office Action, pages 2 and 3.) The Examiner finally asserted that, for claim 1, “[i]t
would have been obvious to one having ordinary skill in the art at the time the invention was
made to use a cache for temporarily store and monitor for improved performance.” (See
25 Office Action, page 3.)

Claim 1

To the extent the Examiner's language at pages 2 and 3 of the Office Action can be
understood, it appears that the Examiner has asserted the following correspondence between
Lipkin and Challenger and claim 1:

Claim 1	<u>Lipkin</u>	<u>Challenger</u>
1. A client-server system capable of validating cached <i>eXtensible Markup Language (XML)</i> data comprising:	-	<u>Challenger</u> does not teach this claim element.
a data store for storing <i>XML</i> data;	-	<u>Challenger</u> does not teach this claim element.
a server for retrieving and updating <i>XML</i> data in the data store to service client requests;	-	<u>Challenger</u> does not teach this claim element.
a transformation engine for transforming <i>XML</i> data into a format suitable for a client application based on a set of transformation rules;	-	<u>Challenger</u> does not teach this claim element.
a cache for temporarily storing transformed <i>XML</i> data as data objects for later reuse;	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.
a cache monitor for ensuring that cached objects are validated when changes to <i>XML</i> data in the data store are detected by the server; and	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.

an object dependency mapper for automatically and continuously determining dependencies between <i>XML</i> data in the data store and sets of transformation rules.	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.
---	--	--

In reviewing the cited portions of Lipkin and Challenger, however, it becomes apparent that Lipkin and Challenger have been generalized, and, in fact, does not support the position asserted by the Examiner.

5 **a cache for temporarily storing transformed *XML* data as data objects for later reuse**

In particular, Lipkin and Challenger, alone or in combination, fail to teach or suggest “a cache for temporarily storing transformed *XML* data as data objects for later reuse”, as required by claim 1. The Examiner admitted that Lipkin fails to teach or suggest “a cache for temporarily storing transformed *XML* data as data objects for later reuse”. (See Office Action, page 2.) Challenger discloses (1) “Local Cache[,which] . . . is a cache (or other standard object store such as a file system) which is updated by an instance of a Trigger Monitor residing on the same physical machine as the cache itself[, and] Remote Cache[, which] . . . is a cache (or other standard object store such as a file system) which is updated by an instance of a Trigger Monitor residing on a different physical machine from the cache itself.” (See Challenger, column 29, lines 52-61.) Thus, Challenger fails to teach or suggest that either the Local Cache or the Remote Cache can be used “for temporarily storing transformed *XML* data as data objects for later reuse”. Therefore, Challenger cannot teach or suggest “a cache for temporarily storing transformed *XML* data as data objects for later reuse”. Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest

the claim 1 element of “a cache for temporarily storing transformed *XML* data as data objects for later reuse”.

**a cache monitor for ensuring that cached objects are validated
when changes to *XML* data in the data store are detected by the server**

5 Also, Lipkin and Challenger, alone or in combination, fail to teach or suggest “a cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”, as required by claim 1. The Examiner admitted that Lipkin fails to teach or suggest “a cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”. (See Office Action, page

10 2.) Challenger discloses “a cache manager 1 (which is an example of an object manager) [that] determines how changes to underlying data affect the values of objects.” (See Challenger, column 8, lines 54-56.) Thus, Challenger fails to teach or suggest that cache manager 1 can be used “for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”. Therefore, Challenger cannot teach or

15 suggest “a cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”. Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest the claim 1 element of “a cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”.

20 **an object dependency mapper for automatically and continuously
determining dependencies between *XML* data in the data store and sets of
transformation rules**

 In addition, Lipkin and Challenger, alone or in combination, fail to teach or suggest “an object dependency mapper for automatically and continuously determining dependencies

25 between *XML* data in the data store and sets of transformation rules”, as required by claim 1. The Examiner admitted that Lipkin fails to teach or suggest “an object dependency mapper for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. (See Office Action, page 2.) Challenger discloses an “object manager [that] maintains the data dependence information . . . [such that] [w]henver

new objects are created or the data dependencies change, the object manager is responsible for updating the appropriate information.” (See Challenger, column 4, lines 33-37.) Thus, Challenger fails to teach or suggest that the object manager can be used “for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. Therefore, Challenger cannot teach or suggest “an object dependency mapper for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest the claim 1 element of “an object dependency mapper for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. It is therefore clear that Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 1 and, therefore, a rejection of claim 1 under 35 U.S.C. § 103(a) would be inappropriate.

Claim 2

Since dependent claim 2 depends on independent claim 1 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 1, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 2, and, therefore, a rejection of claim 2 under 35 U.S.C. § 103(a) is inappropriate.

Claim 3

Since dependent claim 3 depends on dependent claim 2 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 2, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 3, and, therefore, a rejection of claim 3 under 35 U.S.C. § 103(a) is inappropriate.

Claim 4

Since dependent claim 4 depends on dependent claim 3 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 3, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 4, and, therefore, a rejection of claim 4 under 35 U.S.C. § 103(a) is inappropriate.

Claim 5

Since dependent claim 5 depends on dependent claim 4 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 4, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 5, and, therefore, a rejection of claim 5 under 35 U.S.C. § 103(a) is inappropriate.

5 **Claim 6**

Since dependent claim 6 depends on dependent claim 5 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 5, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 6, and, therefore, a rejection of claim 6 under 35 U.S.C. § 103(a) is inappropriate.

10 **Claim 7**

Since dependent claim 7 depends on dependent claim 6 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 6, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 7, and, therefore, a rejection of claim 7 under 35 U.S.C. § 103(a) is inappropriate.

15 **Claims 8 and 12**

Since dependent claims 8 and 12 depend on dependent claim 7 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 7, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 8 or 12, and, therefore, a rejection of claim 8 or 12 under 35 U.S.C. § 103(a) is inappropriate.

20 **Claim 9**

Since dependent claim 9 depends on dependent claim 8 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 8, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 9, and, therefore, a rejection of claim 9 under 35 U.S.C. § 103(a) is inappropriate.

25 **Claims 10 and 11**

Since dependent claims 10 and 11 depend on dependent claim 9 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 9, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 10 or 11, and, therefore, a rejection of claim 10 or 11 under 35 U.S.C. § 103(a) is inappropriate.

Claims 13, 14, and 15

Since dependent claims 13, 14, and 15 depend on dependent claim 12 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 12, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 13, 14, or 15, and, therefore, a rejection of claim 13, 14, or 15 under 35 U.S.C. § 103(a) is inappropriate.

Claims 16-21

The Examiner asserted that, for claim 16,

10 Lipkin discloses substantially all of the elements, a client-server
system capable of validating cached
eXtensible Markup Language (XML) data comprising
a data store for storing XML data;
a server for retrieving and updating XML data in the data store to
15 service client requests;
a transformation engine for transforming XML data into a format
suitable for a client application based on a set of transformation
rules (Paragraph [1190])

20 (See Office Action, page 2.) The Examiner then admitted that, for claim 16,

Lipkin discloses substantially all of the elements . . . except
a cache for temporarily storing transformed XML data as data
objects for later reuse;
25 a cache monitor for ensuring that cached objects are validated
when changes to XML data in the data store are detected by the
server; and
an object dependency mapper for automatically and
continuously determining dependencies between XML data in

the data store and sets of transformation rules.

(See Office Action, page 2.) The Examiner then asserted that, for claim 16,

5 Challenger teaches in figures 1, 4-16, 18-21, 23-28, and 30-42,
a cache for temporarily store, and
monitor the change to data,
an object dependency mapper for automatically and
continuously determining dependencies between data in store
10 and set of transformation rules (Abstract; col. 4, lines 6-10;
col. 10, lines 14-24; col. 29, lines 32 +).

(See Office Action, pages 2 and 3.) The Examiner finally asserted that, for claim 16, “[i]t
would have been obvious to one having ordinary skill in the art at the time the invention was
15 made to use a cache for temporarily store and monitor for improved performance.” (See
Office Action, page 3.)

Claim 16

To the extent the Examiner's language at pages 2 and 3 of the Office Action can be
understood, it appears that the Examiner has asserted the following correspondence between
20 Lipkin and Challenger and claim 16:

<u>Claim 16</u>	<u>Lipkin</u>	<u>Challenger</u>
16. In a client-sever computing system having a cache and storing <i>eXtensible Markup Language (XML)</i> data as data objects, a method for determining invalid cached objects comprising:	-	<u>Challenger</u> does not teach this claim element.

transforming <i>XML</i> data into a format suitable for a client application based on a set of transformation rules;	-	<u>Challenger</u> does not teach this claim element.
determining dependencies between cached objects and <i>XML</i> data related to the cached objects;	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.
monitoring updates to the related <i>XML</i> data; and	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.
determining the cached objects that are affected by changes to the related <i>XML</i> data based on the dependencies.	<u>Lipkin</u> does not teach this claim element.	<u>Challenger</u> does not teach this claim element.

In reviewing the cited portions of Lipkin and Challenger, however, it becomes apparent that Lipkin and Challenger have been generalized, and, in fact, does not support the position asserted by the Examiner.

5 **determining dependencies between cached objects and *XML* data related to the cached objects**

10 In particular, Lipkin and Challenger, alone or in combination, fail to teach or suggest “determining dependencies between cached objects and *XML* data related to the cached objects”, as required by claim 16. The Examiner admitted that Lipkin fails to teach or suggest “an object dependency mapper for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. (See Office Action,

page 2.) Thus, the Examiner admitted that Lipkin fails to teach or suggest “determining dependencies between cached objects and *XML* data related to the cached objects”.

Challenger discloses an “object manager [that] maintains the data dependence information . . . [such that] [w]hen new objects are created or the data dependencies change, the object manager is responsible for updating the appropriate information.” (See Challenger, column 4, lines 33-37.) Thus, Challenger fails to teach or suggest that the object manager can be used for “determining dependencies between cached objects and *XML* data related to the cached objects”. Therefore, Challenger cannot teach or suggest “determining dependencies between cached objects and *XML* data related to the cached objects”. Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest the claim 16 element of “determining dependencies between cached objects and *XML* data related to the cached objects”.

monitoring updates to the related *XML* data

Also, Lipkin and Challenger, alone or in combination, fail to teach or suggest “monitoring updates to the related *XML* data”, as required by claim 16. The Examiner admitted that Lipkin fails to teach or suggest “a cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”. (See Office Action, page 2.) Thus, the Examiner admitted that Lipkin fails to teach or suggest “monitoring updates to the related *XML* data”. Challenger discloses “a cache manager 1 (which is an example of an object manager) determines how changes to underlying data affect the values of objects.” (See Challenger, column 8, lines 54-56.) Thus, Challenger fails to teach or suggest that cache manager 1 can be used for “monitoring updates to the related *XML* data”. Therefore, Challenger cannot teach or suggest “monitoring updates to the related *XML* data”. Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest the claim 16 element of “monitoring updates to the related *XML* data”.

determining the cached objects that are affected by changes to the related *XML* data based on the dependencies

In addition, Lipkin and Challenger, alone or in combination, fail to teach or suggest “determining the cached objects that are affected by changes to the related *XML* data based on

the dependencies”, as required by claim 16. The Examiner admitted that Lipkin fails to teach or suggest “an object dependency mapper for automatically and continuously determining dependencies between *XML* data in the data store and sets of transformation rules”. (See Office Action, page 2.) The Examiner also admitted that Lipkin fails to teach or suggest “a
5 cache monitor for ensuring that cached objects are validated when changes to *XML* data in the data store are detected by the server”. (See Office Action, page 2.) Thus, the Examiner admitted that Lipkin fails to teach or suggest
“determining the cached objects that are affected by changes to the related *XML* data based on the dependencies”. Challenger discloses an “object manager [that] maintains the data
10 dependence information . . . [such that] [w]hen new objects are created or the data dependencies change, the object manager is responsible for updating the appropriate information.” (See Challenger, column 4, lines 33-37.) Challenger also discloses “a cache manager 1 (which is an example of an object manager) determines how changes to underlying data affect the values of objects.” (See Challenger, column 8, lines 54-56.) Thus, Challenger
15 fails to teach or suggest that either the object manager or cache manager 1 can be used for “determining the cached objects that are affected by changes to the related *XML* data based on the dependencies”. Therefore, Challenger cannot teach or suggest “determining the cached objects that are affected by changes to the related *XML* data based on the dependencies”.
Therefore, Lipkin and Challenger, alone or in combination, cannot teach or suggest the claim
20 16 element of determining the cached objects that are affected by changes to the related *XML* data based on the dependencies”. It is therefore clear that Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 16 and, therefore, a rejection of claim 16 under 35 U.S.C. § 103(a) would be inappropriate.

Claims 17, 18, and 19

25 Since dependent claims 17, 18, and 19 depend on independent claim 16 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 16, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 17, 18, or 19 and therefore, a rejection of claim 17, 18, or 19 under 35 U.S.C. § 103(a) is inappropriate.

Claim 20

Since dependent claim 20 depends on dependent claim 19 since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 19, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 20, and therefore, a rejection of claim 20 under 35 U.S.C. § 103(a) is inappropriate.

Claim 21

Since dependent claim 21 depends on dependent claim 20 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 20, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 21, and therefore, a rejection of claim 21 under 35 U.S.C. § 103(a) is inappropriate.

Claims 22-27

Claim 22

Since claim 22, as amended, is the computer program product version of claim 16 with similar elements as claim 16, since the Examiner has asserted arguments against Claim 16 that are similar to the arguments asserted by the Examiner against claim 16, and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 16, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 22 for similar reasons that Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 16, and therefore, a rejection of claim 22 under 35 U.S.C. § 103(a) is inappropriate.

Claims 23, 24, and 25

Since dependent claims 23, 24, and 25 depend on independent claim 22 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 16, Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 23, 24, or 25, and therefore, a rejection of claim 23, 24, or 25 under 35 U.S.C. § 103(a) is inappropriate.

Claim 26

Since dependent claim 26 depends on dependent claim 25 and since Lipkin and Challenger, alone or in combination, cannot teach or suggest each element of claim 25, Lipkin

and Challenger, alone or in combination, cannot teach or suggest each element of claim 26, and therefore, a rejection of claim 26 under 35 U.S.C. § 103(a) is inappropriate.

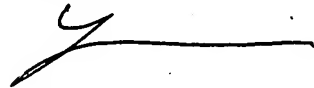
Claim 27

Since dependent claim 27 depends on dependent claim 26 and since Lipkin and
5 Challenger, alone or in combination, cannot teach or suggest each element of claim 26, Lipkin
and Challenger, alone or in combination, cannot teach or suggest each element of claim 27,
and therefore, a rejection of claim 27 under 35 U.S.C. § 103(a) is inappropriate.

CONCLUSION

All the claims presently on file in the present application are in condition for
10 immediate allowance, and such action is respectfully requested. It is respectfully submitted
that the application has now been brought into a condition where allowance of the case is
proper. Reconsideration and issuance of a Notice of Allowance are respectfully solicited.

Respectfully Submitted,



Date: September 12, 2007

Leonard T. Guzman
Reg. No. 46,308

IBM Almaden Research Center
650 Harry Road
C45A/J2B
San Jose, CA 95120

Phone Number: 408-927-3377
Facsimile Number: 408-927-3375

APPENDIX A

CLAIMS APPENDIX

- 5 1. A client-server system capable of validating cached eXtensible Markup Language (XML) data comprising:
- a data store for storing XML data;
- a server for retrieving and updating XML data in the data store to service client requests;
- 10 a transformation engine for transforming XML data into a format suitable for a client application based on a set of transformation rules;
- a cache for temporarily storing transformed XML data as data objects for later reuse;
- a cache monitor for ensuring that cached objects are validated when changes to XML data in the data store are detected by the server; and
- 15 an object dependency mapper for automatically and continuously determining dependencies between XML data in the data store and sets of transformation rules.
2. The system as recited in claim 1 further comprising an object manager for managing data objects in the cache.
- 20 3. The system as recited in claim 2 further comprising a transformation rule alert service for detecting when the transformation rules are modified, added to the system and deleted from the system.
- 25 4. The system as recited in claim 3, wherein the server accesses the object manager to generate a response to a client request for data.
5. The system as recited in claim 4, wherein the server accesses the cache monitor to validate cached objects when a data update request is received.

30

6. The system as recited in claim 5, wherein:
data in the data store is represented as a tree structure having a root node, a plurality of
intermediate nodes and leaf nodes, the leaf nodes representing data in the data store; and
a transformation rule is an expression describing a path from the root node to a
5 particular node in the tree.
7. The system as recited in claim 6, wherein:
a set of the transformation rules constitutes a style sheet; and
the transformation engine receives a style sheet and the data tree as input, and outputs
10 a transformed data object.
8. The system as recited in claim 7, wherein the cache includes a plurality of data objects
each associated with a style sheet used to generate said each object.
- 15 9. The system as recited in claim 8, wherein:
the object manager uses the transformation engine to generate a new object in
response to a client request when the new object does not exist in the cache; and
the object manager stores the new object in the cache automatically.
- 20 10. The system as recited in claim 9, wherein the object manager periodically refreshes
the cache and removes the objects that have been flagged as invalid by the cache monitor.
11. The system as recited in claim 9, wherein the object manager optionally maintains
statistical information for each object in the cache, and automatically removes cached objects
25 that are being accessed infrequently by the clients.
12. The system as recited in claim 7, wherein the object dependency mapper includes a
table of dependencies, each dependency associating a transformation rule with the style sheets
that include the transformation rule.

13. The system as recited in claim 12, wherein the table of dependencies is automatically generated and maintained by the object dependency mapper.
- 5 14. The system as recited in claim 12, wherein the transformation rule alert service communicates updates on the style sheets to the object dependency mapper.
15. The system as recited in claim 12, wherein the cache monitor uses the table of dependencies to determine a set of relevant style sheets, said relevant style sheets referencing
10 a node in the data tree related to a data update; and
the cache monitor accesses the cache to invalidate the objects generated by the relevant style sheets.
16. In a client-sever computing system having a cache and storing eXtensible Markup
15 Language (XML) data as data objects, a method for determining invalid cached objects comprising:
transforming XML data into a format suitable for a client application based on a set of transformation rules;
determining dependencies between cached objects and XML data related to the cached
20 objects;
monitoring updates to the related XML data; and
determining the cached objects that are affected by changes to the related XML data based on the dependencies.
- 25 17. The method as recited in claim 16, wherein the transformed format is html.
18. The method as recited in claim 16, wherein:
a set of transformation rules constitutes a style sheet;
each dependency associates a transformation rule with a style sheet; and

each data object is the data transformed by a style sheet.

19. The method as recited in claim 16, wherein:

data is represented as a tree structure having a plurality of nodes; and

5 the cached objects that are affected by the data changes are determined using the tree structure.

20. The method as recited in claim 19, wherein the dependencies are maintained in a table of dependencies.

10

21. The method as recited in claim 20, wherein the step of determining the affected objects comprises:

identifying the nodes associated with data updates;

identifying the transformation rules related to the identified nodes;

15 determining a set of relevant style sheets using the table of dependencies, the relevant style sheets including the identified transformation rules; and

identifying the cached objects that have been transformed by the relevant style sheets.

22. In a client-server computing system having a cache and storing eXtensible Markup

20 Language (XML) data as data objects, a computer program product for determining invalid cached objects comprising:

a computer readable medium;

means, provided on the computer readable medium, for transforming XML data into a format suitable for a client application based on a set of transformation rules;

25 means, provided on the computer readable medium, for determining dependencies between cached objects and XML data related to the cached objects;

means, provided on the computer readable medium, for monitoring updates to the related XML data; and

means, provided on the computer readable medium, for determining the cached objects that are affected by changes to the related XML data based on the dependencies.

23. The computer program product as recited in claim 22, wherein the transformed format is html.

24. The computer program product as recited in claim 22, wherein:
a set of transformation rules constitutes a style sheet;
each dependency associates a transformation rule with a style sheet; and
each data object is the data transformed by a style sheet.

25. The computer program product as recited in claim 22, wherein:
data is represented as a tree structure having a plurality of nodes; and
the cached objects that are affected by the data changes are determined using the tree structure.

26. The method as recited in claim 25, wherein the dependencies are maintained in a table of dependencies.

27. The computer program product as recited in claim 26, wherein the means for determining the affected cached objects comprises:

means, provided on the computer readable medium, for identifying the nodes associated with data updates;

means, provided on the computer readable medium, for identifying the transformation rules related to the identified nodes;

means, provided on the computer readable medium, for determining a set of relevant style sheets using the table of dependencies, the relevant style sheets including the identified transformation rules; and

means, provided on the computer readable medium, for identifying the cached objects that have been transformed by the relevant style sheets.

APPENDIX B

EVIDENCE APPENDIX

- 5 There is no applicable evidence.

APPENDIX C

RELATED PROCEEDINGS APPENDIX

- 5 There are no related proceedings.